C Sharp Programming Exercise 02
Recursive Methods

C# Step by Step

This activity consists of four programming exercises. The following exercises are open book and open note. You are free to use any written documentation you wish. However, these are individual exercises, and you cannot consult with each other in writing your programs.

This programming exercise has four parts consisting of four requirements. The grade for each requirement is indicated, for a maximum of 100 points. At a minimum, your program must compile successfully and run.

A *recursive* function is a function that calls itself. Recursive activities are extremely common in every day life. When you mow your lawn, you cut one mower-width and see if you are finished. If so, you stop and drink a beer. If not, you cut one mower-width and see if you are finished, again and again. When you shop for groceries, you place an item in your cart and mark it off the list. If it's the last item, you check out. If not, you place an item in your cart and mark it off the list, again and again. Recursive functions work exactly the same way. You call the function, perform one task, and check to see if you are done. If so, you stop. If not, you call the function again. I have completed the first part of this exercise below, and you can see how simple this is.

Warm up Create a console application that will accept ten numbers between 0 and 100, and report their sum. The program below illustrates one way in which this can be done. Create this program and make sure you understand how it works.

```
namespace sum2ten
{
    class Program
    {
        static void Main(string[] args)
        {
            int start = 0;
            int end = 10;
            int sum = 0;
            sum = get_sum(start, end, sum);
            Console.WriteLine($"The sum is {sum}");
        }

        private static void get_sum(int start, int end, int sum)
        {
            Console.WriteLine($"get_sum({start}, {end}, {sum})");
            start = start + 1;
            sum = sum + start;
            if (start < end)
                return(get\_sum(start, end, sum));
            else
                return(sum);
        }
    }
}
```

Setup and assign letter grades: 70 points Create a method that will accept a numeric (integer, float, or double) argument and return a letter grade: 90 or better is A, 80 or better is B, 70 or better is C, 60 or better is D, and anything less than 60 is F. You will use this method to assign letter grades in the next parts of this exercise.

Average ten scores: 80 points Create a console application that will accept ten test scores between 0 and 100, average them, and report the numerical grade. For example, a teacher will input ten test scores and compute the average numerical grade. Assign a letter grade to the student.

Average a specific number of scores: 90 points Create a console application that will accept an arbitrary number test scores (as specified by the user) between 0 and 100, average them, and report a numerical grade for the average. For example, a teacher will input the total number of tests, then input the specified number of test scores and compute the average numerical grade. Assign a letter grade to the student.

Average a non-specific number of scores: 100 points Create a console application that will accept a number test scores (as calculated by the number of scores actually entered) between 0 and 100, average them, and report a letter grade for the average based on the usual scale. For example, a teacher will input any number test scores, and compute the average numerical grade and the letter grade.This part required you to program a *stop value*. You can choose any kind of stop value you want, typically stop values consist of "quit," "exit," Ctl-C, or Escape. I chose a negative one (-1) as a stop value, assuming that no student would ever score a negative grade on a test. Assign a letter grade to the student.

Getting started Here is a template that will help you get started.

```
namespace progex02
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("\nPart 1, sum 10 numbers.");
            int sum = SumTenInts(0, 0);
            char letterGrade = 'X';
            Console.WriteLine($"The sum of ten integers is {sum}");

            Console.WriteLine("\nPart 2, average 10 numbers.");
            double avg = AvgTenInts(0, 0);
            letterGrade = ConvertNumericToLetterGrade(avg)
            Console.WriteLine($"The average of ten integers is {avg} and the letter grade is
                 {letterGrade}");

            Console.WriteLine("\nPart 3, average user predetermined number of scores.");
            Console.Write("How many scores do you wish to enter? ");
            string noScores = Console.ReadLine();
            int numScores = int.Parse(noScores);
            double avg1 = AvgUnkInts(0, 0, numScores);
            letterGrade = ConvertNumericToLetterGrade(avg1)
            Console.WriteLine($"The average of {numScores} integers is {avg1} and the letter
                 grade is {letterGrade}");

            Console.WriteLine("\nPart 4, average non-predetermined number of scores.");
            double avg2 = AvgAnyInts(0, 0);
            letterGrade = ConvertNumericToLetterGrade(avg2)
            Console.WriteLine($"The average of ten integers is {avg2} and the letter grade
                is {letterGrade}");
        }

        private static char ConvertNumericToLetterGrade(double grade)
        {
            /*some code*/
        }

        private static double AvgAnyInts(int sum, int count)
        {
            /*some code*/
        }

        private static double AvgUnkInts(int sum, int count, int numScores)
        {
            /*some code*/
        }

        private static double AvgTenInts(int sum, int count)
        {
            /*some code*/
        }

        private static int SumTenInts(int sum, int count)
        {
            Console.Write("Enter a score: ");
            string input = Console.ReadLine();
            sum += int.Parse(input);
            if (count < 10)                     // recursive call
                return SumTenInts(sum, count + 1);
            else                                // end case
                return sum;
        }
    }
```